



**HAL**  
open science

# Improving Complexity Bounds for the Computation of Puisseux Series over Finite Fields

Adrien Poteaux, Marc Rybowicz

► **To cite this version:**

Adrien Poteaux, Marc Rybowicz. Improving Complexity Bounds for the Computation of Puisseux Series over Finite Fields. ISSAC '15, Jul 2015, Bath, United Kingdom. pp.299–306, 10.1145/2755996.2756650 . hal-01253216

**HAL Id: hal-01253216**

**<https://hal.science/hal-01253216>**

Submitted on 8 Jan 2016

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Improving Complexity Bounds for the Computation of Puiseux Series over Finite Fields

Adrien Poteaux  
CRISTAL

UMR 9189 Université de Lille 1 - CNRS  
Bâtiment M3  
59655 Villeneuve d'Ascq, France  
adrien.poteaux@univ-lille1.fr

Marc Rybowicz  
XLIM-DMI

UMR 7252 Université de Limoges - CNRS  
123 avenue Albert Thomas  
87060 Limoges Cedex  
marc.rybowicz@unilim.fr

Let  $L$  be a field of characteristic  $p$  with  $q$  elements and  $F \in L[X, Y]$  be a polynomial with  $p > \deg_Y(F)$  and total degree  $d$ . In [40], we showed that rational Puiseux series of  $F$  above  $X = 0$  could be computed with an expected number of  $\mathcal{O}(d^5 + d^3 \log q)$  arithmetic operations in  $L$ . In this paper, we reduce this bound to  $\mathcal{O}(d^4 + d^2 \log q)$  using Hensel lifting and changes of variables in the Newton-Puiseux algorithm that give a better control of the number of steps. The only asymptotically fast algorithm required is polynomial multiplication over finite fields. This approach also allows to test the irreducibility of  $F$  in  $\overline{L}[[X]][Y]$  with  $\mathcal{O}(d^3)$  operations in  $L$ . Finally, we describe a method based on structured bivariate multiplication [34] that may speed up computations for some input.

## 1 Introduction

Let  $L$  be a field of characteristic  $p$  with  $q$  elements,  $\overline{L}$  be an algebraic closure of  $L$  and  $F$  be a polynomial in  $L[X, Y]$  with partial degrees  $\deg_X(F) = d_X > 0$ ,  $\deg_Y(F) = d_Y > 0$  and total degree  $\deg(F) = d$ . We assume that  $F$ , considered as a polynomial in  $Y$ , is separable and primitive, hence squarefree and without non trivial factor in  $L[X]$ .

We also assume in the sequel that  $p > d_Y$ . Therefore, for any  $x_0 \in \overline{L}$ , it is well-known that the roots of  $F$  may be expressed as fractional Laurent power series in  $(X - x_0)$  with coefficients in  $\overline{L}$ , called (*classical*) *Puiseux series of  $F$  above  $x_0$*  (CPS in the sequel). Terms written *in italics* in this introduction will be defined in section 2. If  $p \leq d_Y$ , CPS may not exist and other types of expansions are necessary: generalized Puiseux series, see [29] and references therein, or Hamburger-Noether expansions [10, 43].

CPS are an important tool to study singularities of the curve  $F(X, Y) = 0$  [9, 54], to determine the genus of the curve via Riemann-Hurwitz's formula, to determine bases of Riemann-Roch spaces [22, 7], to compute integral bases of the function field  $L(X)[Y]/(F)$  [50], etc. Code for computing CPS is available for instance in Maple [36] (implemented by Van Hoeij [50]), Magma [8] (see Beck [4]) or Singular [19] (implemented by Lamm and Lossen, see [26]).

Our interest in the finite field case stemmed from a modular reduction method that we proposed to avoid coefficient swell in the number field case [39, 41]. In this context, the condition  $p > d_Y$  may always be enforced and is part of our good reduction criterion. Our goal is to compute *singular parts* of CPS since they contain the arithmetic and geometric information required for most applications. When singular parts are known, subsequent terms of CPS may be efficiently computed using quadratic Newton iterations [31].

It is however more convenient to compute singular parts of *rational Puiseux expansions* (RPE) of  $F$  rather than CPS. Introduced by Duval [22, 23], RPEs allow to work in the *residue fields* of the *places* of the function field  $L(X)[Y]/(F)$  (or the product of function fields if  $F$  is not irreducible in  $L[X, Y]$ ). Computations therefore take place in optimal degree extensions of  $L$  and RPEs provide arithmetical insight. CPS may easily be recovered from RPEs; see Section 2.

In [40], we studied how to truncate coefficients throughout the computations and gave a detailed count of the number of arithmetic operations in  $L$  required by Duval's version of the Newton-Puiseux algorithm to compute RPEs.

Inspired by a proof by Abhyankar of Newton-Puiseux's Theorem [1, Chapter 12], we show herein that it is possible to improve this result using Hensel-like factorizations and simple, but essential, substitutions.

We explain precisely our goal in Section 2 and recall main features of the Rational Newton-Puiseux algorithm in Section 3. We describe the improved algorithm in Section 4 and study its worst case complexity in Section 5. Finally, we show that if a fast multiplication algorithm for bivariate polynomials with support in a lattice is given (see [34, Theorem 12]), then the input polynomials may be factorized at an acceptable cost. This modification may lead to improved performances for some families of input, but does not reduce worst case asymptotic complexity.

The main contributions of this paper are Theorem 1 and 2 below. Notation  $\mathcal{O}$  hides logarithmic factors of the degrees:

**Theorem 1.** *There is an algorithm to compute singular parts of a system of RPEs above 0 of  $F$  with an expected number of  $\mathcal{O}(d_X d_Y^3 + d_Y^2 \log \mathfrak{q}) \subset \mathcal{O}(d^4 + d^2 \log \mathfrak{q})$  field operations in  $L$ .*

This result should be compared with the bound  $\mathcal{O}(d^5 + d^3 \log \mathfrak{q})$  given in [40], where we also derived an  $\mathcal{O}(d^5 \log \mathfrak{q})$  bound for the computation of the genus of the curve defined by  $F$  and new complexity results for the number field case. Unfortunately, the improvement given by Theorem 1 does not propagate to genus computation; we will discuss this issue in the conclusion.

**Theorem 2.** *There is an algorithm to decide whether  $F$  is irreducible in  $\overline{L}[[X]][Y]$  performing  $\mathcal{O}(d_X d_Y^2) \subset \mathcal{O}(d^3)$  operations in  $L$ .*

**Related works.** The complexity of the Newton-Puiseux algorithm, in its classical or rational form, has been investigated by Chistov [12], Duval [23], and Walsh [56, 55]. Other approaches to compute CPS have been proposed: linear algebra [21] (following [15]) and differential equations [16, 13, 14, 47, 49, 17], notably. Merle and Henry [27], then Teitelbaum [46] studied the arithmetic complexity of the resolution of the singularity at the origin defined by  $F(X, Y) = 0$ , a process tightly related to Puiseux series [9]. We have commented on these works and explained why we prefer to stick to the Newton-Puiseux algorithm in [40, 41].

Sasaki and als. use generalizations of Hensel's lifting to compute Puiseux series [45] or for polynomial factorization in  $\overline{L}[[X]][Y]$  [44, 28], but no complexity analysis is given. In [32], Kuo revisited the theory of algebroids in  $\mathbb{C}[[X, Y]]$ , avoiding CPS and singularity resolution processes, and gave an irreducibility test in  $\mathbb{C}[[X, Y]]$  [33] that could probably be extended to finite fields, but did not demonstrate that his approach is competitive. More recently, Berthomieu, Lecerf and Quintin [5, Section 3] proposed a Hensel-like factorization method to speed up the computation of roots of  $F$  in  $L[[X]]$ ; with our notations, they obtain an  $\mathcal{O}(d^3)$  algorithm, thus gaining an order of magnitude.

Factorization of  $F$  in  $L[[X]][Y]$  or  $\overline{L}[[X]][Y]$  is closely related to CPS since minimal polynomials over  $L$  or  $\overline{L}$  of CPS are the irreducible factors of  $F$ . The factorization of univariate polynomials over local fields, such as  $L((X))$ , has been studied intensively; see [37, 38, 25, 3] and references therein. In particular, the Montes algorithm has received a lot of attention recently. Bauch, Nart and Stainsby [3] have proved that the factorization of a monic  $F$  over  $L[[X]][Y]$  up to precision  $\mu$  can be achieved in  $\mathcal{O}(d_Y^2 + d_Y \mathcal{V}_F^2 + d_Y(1 + \mathcal{V}_F) \log \mathfrak{q} + d_Y^2 \mu)$  operations in  $L$ , where  $\mathcal{V}_F = v_X(\Delta_F)$  is the valuation of the discriminant of  $F$ . In our context, we may set  $\mu = \mathcal{V}_F$  (see [40, Section 4]) and remark that  $\mathcal{V}_F \in \mathcal{O}(d_X d_Y)$ . This yields  $\mathcal{O}(d^5 + d^3 \log \mathfrak{q})$ , as in [40]. They also provide an irreducibility test that runs in  $\mathcal{O}(d_Y^2 + d_Y(1 + \mathcal{V}_F) \log \mathfrak{q} + \mathcal{V}_F^2) \subset \mathcal{O}(d^4 + d^3 \log \mathfrak{q})$ . For genus computation, [2] proposed a method with an  $\mathcal{O}(d^7 \log \mathfrak{q})$  complexity bound, but more promising experimental results. Algorithms derived from Montes' method have so far not demonstrated a better asymptotic complexity than the classical Newton-Puiseux approach for  $L((X))$ . Besides, they are significantly more involved.

### Additional notations and definitions.

- For  $S \in \overline{L}[[X]]$ , we denote by  $v_X(S)$  the  $X$ -adic valuation of  $S$  and extend this notation to fractional power series.
- For  $t \in \mathbb{N}^*$ ,  $L_t$  is the degree  $t$  extension of  $L$  in  $\overline{L}$ .
- If  $S = \sum_k \alpha_k X^{k/e}$  is a fractional power series in  $\overline{L}((X^{1/e}))$  and  $r$  is a rational number,  $[S]^r$  denotes the truncated series  $[S]^r = \sum_{k \leq N} \alpha_k X^{k/e}$  where  $N = \max\{k \in \mathbb{N} \mid \frac{k}{e} \leq r\}$ . It is extended to elements of  $\overline{L}((X^{1/e}))[Y]$  coefficient-wise.
- For  $e \in \mathbb{N}^*$ ,  $\zeta_e$  is a primitive  $e$ -th root of unity.

- For a polynomial  $H = \sum a_{ij}X^jY^i \in \overline{L}[X, Y]$ :
- $d_Y(H)$  is its degree with respect to  $Y$ ,
- $\Delta_H$  is its discriminant with respect to  $Y$ ,
- $\mathcal{V}_H = v_X(\Delta_H)$ ,
- $\mathcal{I}(H) = v_Y(H(0, Y))$ ,
- $\text{Supp}(H) = \{(i, j) \in \mathbb{N}^2 \mid a_{ij} \neq 0\}$  is the *support* of  $H$ ,
- $H$  is called  *$Y$ -monic* if it is monic in the variable  $Y$ .

**Complexity model and arithmetic cost.** To estimate algorithm complexity, we just count the number of operations (addition, multiplication, division) in  $L$ . Multiplying by the binary cost of operations in  $L$  should give realistic bounds for running times of a careful implementation. Our algorithm is deterministic and we consider worst case estimates. However, it makes use of sub-algorithms for factoring polynomials in  $L[T]$  and computing primitive elements in finite fields that are probabilistic of Las Vegas type. For them, we use upper bounds for average arithmetic complexity that propagate to Theorem 1.

Our complexity results require asymptotically fast algorithms for polynomial arithmetic and we will use bounds below for basic task arithmetic complexity. When no specific reference is given, the result may be found in [53]. Integers  $n_X$  and  $n_Y$  are bounds for degrees in  $X$  and  $Y$  of input polynomials.

- Multiplication of two polynomials in  $L[X]$ :  $\mathcal{O}(n_X)$ .
- Multiplication of two polynomials in  $L[X, Y]$ :  $\mathcal{O}(n_X n_Y)$ .
- Operations in  $L_t$ :  $\mathcal{O}(t)$  with primitive representation.
- Factorization of a polynomial in  $L[X]$ :  $\mathcal{O}(n_X^2 + n_X \log q)$ .
- Computation of  $\Delta_H$ :  $\mathcal{O}(n_X n_Y^2)$ .
- $Y$ -shift, i.e. computation of  $H(X, Y + B) \bmod X^{n_X+1}$  where  $H \in L[X, Y]$  is  $Y$ -monic and  $B$  is in  $L[X]$ :  $\mathcal{O}(n_X n_Y)$  if  $n_X > 0$  and  $\mathcal{O}(n_Y)$  if  $n_X = 0$ . Indeed, if  $p > n_Y$ , [6, Problem 2.6] shows how to perform a shift in a univariate polynomial of degree  $n_Y$  with coefficients in a commutative ring  $\mathbb{A}$  with  $\mathcal{O}(n_Y)$  operations in  $\mathbb{A}$ . Taking  $\mathbb{A} = L[X]/(X^{n_X+1})$  gives the above bounds.

## 2 Rational Puiseux Expansions

In this section, we precisely set our goal and recall useful properties. Let  $L$  and  $F = \sum_{l=0}^{d_Y} A_l(X)Y^l$  be as in Section 1. Up to a change of variable  $X \mapsto X + x_0$  and an extension of the ground field  $L$ , it is sufficient to give definitions and properties for the case  $x_0 = 0$ . Following Duval [23], we consider decompositions into irreducible elements:

$$F = \prod_{i=1}^{\rho} F_i \text{ with } F_i \text{ irreducible in } L[[X]][Y] \quad (1)$$

$$F_i = \prod_{j=1}^{f_i} F_{ij} \text{ with } F_{ij} \text{ irreducible in } \overline{L}[[X]][Y] \quad (2)$$

$$F_{ij} = A_{d_Y} \prod_{k=0}^{e_i-1} \left( Y - S_{ij}(X^{1/e_i} \zeta_{e_i}^k) \right) ; S_{ij} \in \overline{L}[[X]] \quad (3)$$

**Definition 1.** The series  $S_{ijk}(X) = S_{ij}(X^{1/e_i} \zeta_{e_i}^k)$  are the classical Puiseux series (CPS) of  $F$  above 0.

**Proposition 1.** The  $\{F_{ij}\}_{1 \leq j \leq f_i}$  have coefficients in a finite extension  $K_i$  of  $L$  and  $f_i = [K_i : L]$ . They are conjugated by the action of the Galois group of  $K_i/L$ .

**Definition 2.** A system of rational Puiseux expansions over  $L$  ( $L$ -RPE) of  $F$  above 0 is a set  $\{R_i\}_{1 \leq i \leq \rho}$  such that:

- $R_i(T) \in K_i((T))^2$ ,
- $R_i(T) = (X_i(T), Y_i(T)) = (\gamma_i T^{e_i}, \sum_{l=n_i}^{\infty} \beta_{il} T^l)$ ,  $\gamma_i \neq 0$ ,
- $R_i$  is a parametrization of  $F_i$ , i.e.  $F_i(X_i(T), Y_i(T)) = 0$ ,
- the parametrization is irreducible, i.e.  $e_i$  is minimal.

If  $Y_i(0)$  is defined,  $(X_i(0), Y_i(0))$  is called the center of  $R_i$ .

Duval [23] showed that there is a canonical bijection between the  $R_i$  and the *places over  $L$*  (see [11, 24] for a definition) of the algebraic function fields defined by the irreducible factors of  $F$  in  $L[X, Y]$ . Under this correspondence, residue fields of the places are isomorphic to the coefficient fields of the  $R_i$  and ramifications indices of the places are equal to the  $e_i$ . This leads to the following terminology:

**Definition 3.** *The integer  $e_i$  is the ramification index of  $R_i$ ,  $K_i$  is its residue field and  $f_i$  its residual degree.*

**Proposition 2.**  $\sum_{i=1}^{\rho} e_i f_i = d_Y$ .

From a system of  $L$ -RPEs, CPS can easily be recovered:

1.  $R_i$  has  $f_i$  conjugates  $R_{ij}(T) = (X_{ij}(T), Y_{ij}(T))$  over  $L$ :

$$X_{ij} = \gamma_{ij} T^{e_i} \text{ and } Y_{ij} = \sum_{l=n_i}^{\infty} \beta_{ijl} T^l ; 1 \leq j \leq f_i$$

2. The CPS are  $S_{ijk}(X) = Y_{ij} \left( \zeta_{e_i}^k X^{1/e_i} / \gamma_{ij}^{1/e_i} \right)$ , where  $\gamma_{ij}^{1/e_i}$  denotes any  $e_i$ -th root of  $\gamma_{ij}$  and  $0 \leq k \leq e_i - 1$ .

**Definition 4.** *Define  $s_i = \min \{0, n_i\}$ . The regularity index  $r_i$  of  $S_{ijk}$  in  $F$  is the least integer  $N \geq s_i$  such that  $[S_{ijk}]_{e_i}^N = [S_{uvw}]_{e_i}^N$  implies  $(u, v, w) = (i, j, k)$ . The truncated series  $[S_{ijk}]_{e_i}^{\frac{r_i}{e_i}}$  is the singular part of  $S_{ijk}$  in  $F$ . The regularity index  $r_i$  of  $R_i$  in  $F$  is that of  $S_{ijk}$  in  $F$ , for any  $j, k$ . The singular part of  $R_i$  is  $\left( \gamma_i T^{e_i}, \sum_{l=n_i}^{r_i} \beta_{ik} T^l \right)$ .*

In other words, the regularity index of  $S_{ijk}$  is the smallest truncation order that allows to distinguish  $S_{ijk}$  from other CPS of  $F$ ; see [40, page 194]. Singular parts contain arithmetic and geometric information necessary for many applications: ramification indices, residual degrees, Puiseux exponents, etc. We aim at computing them efficiently.

### 3 Newton-Puiseux Algorithm

This work improves a version of Duval's rational Newton-Puiseux algorithm presented in [40], called **RNPuiseux**. In this section, we introduce some notations and recall useful facts regarding **RNPuiseux**. Because of space constraints, the reader is referred to [40, Section 3] for a detailed description of this algorithm.

**Definition 5.** *The Newton polygon  $\mathcal{N}(H)$  of a polynomial  $H$  in  $L_t[X, Y]$  is the lower part of the convex hull of its support.*

If  $\text{Supp}(H)$  is a vertical line,  $\mathcal{N}(H)$  is reduced to a point. Otherwise,  $\mathcal{N}(H)$  is a sequence of (non degenerate) edges with increasing slopes. In order to get exactly singular parts of RPEs and no superfluous terms, it is convenient to modify slightly this definition (see examples of Figure 1):

**Definition 6.** *Let  $H = \sum_i A_i(X) Y^i$  be squarefree, primitive, with  $d_Y(H) > 0$ . The modified Newton polygon<sup>1</sup>  $\mathcal{N}^*(H)$  is constructed as follow: If  $A_0 = 0$  (resp.  $A_0 \neq 0$  and the first edge, starting from the left, ends at  $(1, v_X(A_1))$ ), add to  $\mathcal{N}(H)$  (resp. replace the first edge by) a fictitious edge joining the vertical axis to  $(1, v_X(A_1))$  such that its slope is the largest (negative or null) integer less than or equal to the slope of the next edge.*

The introduction of  $\mathcal{N}^*$  is motivated by the next example:

**Example 1.** *Consider  $F(X, Y) = (Y - X^k)(Y^2 - X^3)$  with  $k \geq 3$ . CPS of  $F$  are  $S_1 = X^k$ ,  $S_{2,j} = (-1)^j X^{3/2}$ ,  $j = 1, 2$ . According to Definition 4, regularity indices are respectively  $r_1 = 2$  and  $r_{2,j} = 3$ . Using  $\mathcal{N}(F)$  would cause the algorithm to return  $X^k$  for the singular part of  $S_1$ , instead of the expected value  $[S_1]_2^2 = 0$ . If a dense representation is used for the output, returning  $X^k$  would not allow to bound running times in terms of  $\mathcal{V}_F$  (see Proposition 9) because  $\mathcal{O}(k)$  operations would be required to build the result, while  $\mathcal{V}_F = 9$  for any  $k > 1$ .*

Each edge  $\Delta$  of  $\mathcal{N}^*(H)$  corresponds to three integers  $q, m$  and  $l$  with  $q > 0$ ,  $q$  and  $m$  coprime, such that  $\Delta$  is on the line  $qj + mi = l$ . If  $\Delta$  is an horizontal edge,  $m = l = 0$  and we choose  $q = 1$ .

<sup>1</sup> $\mathcal{N}^*(H)$  is more convenient herein than the generic Newton polygon used in [40] and yields essentially the same output.

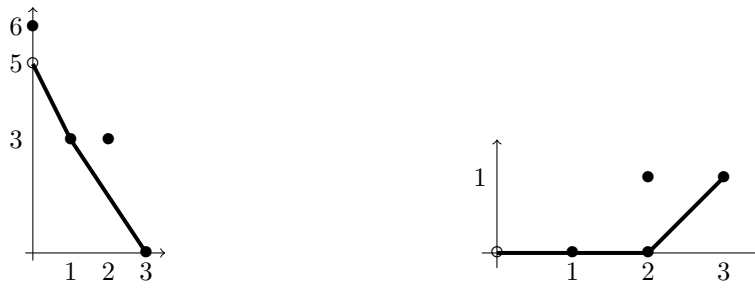


Figure 1:  $\mathcal{N}^*(H)$  for  $(Y - X^3)(Y^2 - X^3)$  and  $Y(Y - 1)(XY - 1)$

**Definition 7.** If  $H = \sum a_{ij}X^jY^i$ , then the characteristic polynomial  $\phi_\Delta$  of  $\Delta$  is  $\phi_\Delta(T) = \sum_{(i,j) \in \Delta} a_{ij}T^{\frac{i-j_0}{q}}$  where  $i_0$  is the smallest value such that  $(i_0, j_0)$  belongs to  $\Delta$  for some  $j_0$ . In particular,  $\phi_\Delta(T) = T$  if  $\Delta$  is a fictitious edge.

When applied to  $(L, F, \mathcal{V}_F)$ , **RNPuiseux** returns a set of pairs<sup>2</sup>  $\mathcal{R}_t(F) = \{(P_i, Q_i)\}_{1 \leq i \leq \rho}$  representing singular parts of  $L$ -RPEs of  $F$  above 0. More precisely:

- $P_i \in \overline{L}[X]$  is a monomial of the form  $\lambda_i X^{e_i}$ ,
- $Q_i(X, Y) = Q_{i0}(X) + c_i Y X^{r_i}$ , where  $(P_i(T), Q_{i0}(T))$  is the singular part of an RPE of  $F$ ,  $r_i$  its regularity index, and  $c_i \in K_i$ .

Starting at  $H = F$ , algorithm **RNPuiseux** consists in recursive applications of transformations:

$$H_{\Delta, \xi}(X, Y) = H(\xi^b X^q, X^m(\xi^a + Y))/X^l \quad (4)$$

where integers  $(q, m, l)$  are determined by an edge  $\Delta$  of  $\mathcal{N}^*(H)$ ,  $\xi$  is a root of  $\phi_\Delta$  and  $a$  and  $b$  are integers satisfying  $aq - bm = 1$  and  $0 \leq b < q$ . These transformations are applied for each relevant pair  $(\Delta, \xi)$  and the algorithm is called recursively on  $H_{\Delta, \xi}$  until  $\mathcal{I}(H) = 1$ , yielding a computation tree whose nodes and leaves are **RNPuiseux** function calls. It is shown in [40] that the expected number of operations in  $L$  required by **RNPuiseux** is in  $\mathcal{O}(d_X^2 d_Y^3 + d_Y^2 d_X \log q)$ .

The following remark and lemma are essential for understanding the next sections:

**Remark 1.** To compute all RPEs of  $H$  above 0, it is sufficient to compute RPEs centered at  $(0, 0)$  of the  $H_{\Delta, \xi}$ . Consequently, for the initial call, (i.e.  $H = F$ ) all edges  $\Delta$  are considered, but recursive calls of **RNPuiseux** treats only edges with negative slopes.

## 4 Improving RNPuiseux

To simplify the exposition, from now on, we assume that **the input polynomial  $F$  is  $Y$ -monic**, but this section may easily be adapted to non monic  $F$  as in [40].

Our improvements rely on the following observations:

- Consider first the obvious following consequence of Weierstrass Preparation Theorem:

**Proposition 3.** If  $G \in L_t[X, Y]$  satisfies  $\mathcal{I}(G) > 0$ , then there exist unique  $\widehat{G}$  and  $U$  in  $L_t[[X]][Y]$  such that:

- $G = \widehat{G}U$
- $U(0, 0) \neq 0$ , i.e.  $U$  is a unit in  $L_t[[X, Y]]$ ,
- $\widehat{G}$  is monic, with  $d_Y(\widehat{G}) = \mathcal{I}(G)$ ,

Moreover, RPEs of  $G$  and  $\widehat{G}$  centered at  $(0, 0)$  are the same.

Polynomial  $\widehat{G}$  is called the *distinguished polynomial* associated with  $G$ . In view of Remark 1, we can replace  $H_{\Delta, \xi}$  in **RNPuiseux** by an approximation  $\widetilde{H}_{\Delta, \xi}$  of its distinguished polynomial  $\widehat{H}_{\Delta, \xi}$ , provided that we can compute an approximation that preserves singular parts of RPEs at a sufficiently low cost; see Proposition 4 below. This will ensure that the input polynomial  $H$  of our algorithm is always monic with  $d_Y(H) = \mathcal{I}(H)$  and that edges of  $\mathcal{N}^*(H)$

<sup>2</sup>In [40], **RNPuiseux** actually returns triplets  $(G_i, P_i, Q_i)$  but  $G_i$  is useless for our purpose.

have negative slopes, except maybe for the initial function call. Moreover, degrees of input polynomials for recursive function calls will be lower.

- Assuming that the above factorization step is performed, it is possible to get a better control on the number of recursive calls and, in particular, to ensure that the sequence of integers  $d_Y(H)$  along a branch of the computation tree is strictly decreasing. Degrees are stationary, i.e.  $d_Y(H) = d_Y(H_{\Delta, \xi})$ , if and only if  $H$  has a single edge  $\Delta$  with  $\phi(\Delta) = (T - \xi)^{d_Y(H)}$ . In this case,  $\Delta$  has integer slope (i.e.  $q = 1$ ) and all Puiseux series of  $H$  have a first term equal to  $\xi X^m \in L_t[X]$ . Following Abhyankar [1, Chapter 12], we propose to use a simple trick to avoid this case: just remove at once all common polynomial terms of Puiseux series by replacing  $H$  with  $\overline{H} = H(X, Y - A_{n_Y-1}/n_Y)$ , where  $n_Y = d_Y(H)$ . Then, if  $\mathcal{N}^*(\overline{H})$  still has a unique edge  $\overline{\Delta}$  with integer slope,  $\phi_{\overline{\Delta}}$  cannot be of the form  $(T - \xi)^{d_Y(H)}$  because its monomial of degree  $d_Y(H) - 1$  is null and  $p > d_Y(H)$ . Therefore, the algorithm will always split  $\overline{H}$  and degrees will be reduced for subsequent recursive calls.

**Remark 2.** *It is worth noting the following: factorization steps alone do not suffice to reduce **RNPuiseux** complexity, but they allow to apply the method above to decrease the number of recursive calls and get a more accurate count of arithmetic operations.*

We now specify sub-algorithms and describe a recursive version of the main algorithm **ARNP**, wherein we emphasize simplicity rather than efficiency. The first one is just an application of Hensel lifting as described in [53, Chapter 15] to get an effective version of Weierstrass Preparation Theorem.

**Proposition 4.** *Let  $G \in L_t[X, Y]$  be a polynomial satisfying hypotheses of Proposition 3,  $N$  be in  $\mathbb{N}$  and  $\widehat{G}$  denote the distinguished polynomial of  $G$ . There exists an algorithm **WPT** such that  $\text{WPT}(L_t, G, N)$  computes  $\widetilde{G} = \lceil \widehat{G} \rceil^N$  with  $\mathcal{O}(Nd_Y(G))$  operations in  $L_t$ .*

**Proposition 5.** *Let  $\phi$  be in  $L_t[T]$  with  $n = d_T(\phi)$ . There exists an algorithm **Oneroot** such that  $\text{Oneroot}(L_t, \phi)$  decides if  $\phi(T) = c(T - \xi)^n$  for some  $c, \xi \in L_t$  and computes  $\xi$  if the answer is positive using  $\mathcal{O}(n)$  operations in  $L_t$ .*

**Bézout**( $q, m$ )

**Input:**  $(q, m) \in \mathbb{Z}^2$  with  $q > 0$ .

**Output:**  $(a, b) \in \mathbb{Z}^2$  such that  $aq - bm = 1$  and  $0 \leq b < q$ .

**Factor**( $L_t, \phi$ )

**Input:**  $L_t$ , a field, and  $\phi \in L_t[T]$ , with  $d_T(\phi) > 0$ .

**Output:** A set  $\{(\phi_i, M_i)\}_i$  with  $\phi_i$  monic, irreducible in  $L_t[T]$  and  $\phi = c \prod_i \phi_i^{M_i}$  for some  $c \in L_t$ .

**ARNP**( $L_t, H, N$ )

**Input:**  $L_t$ , a field,  $H = \sum_i A_i(X)Y^i \in L_t[X, Y]$ , separable,  $Y$ -monic, with  $\deg_Y(H) > 0$ ,  $N \in \mathbb{N}$  (truncation order).

**Output:** If  $N$  is large enough,  $\mathcal{R}_t(H)$ .

1. If  $d_Y(H) = 1$  then Return  $\{[X, Y]\}$
2. If  $\mathcal{N}^*(H)$  is made of a unique edge  $\Delta$  with integer slope and  $\text{Oneroot}(L_t, \phi_\Delta)$  then
3.  $B \leftarrow A_{d_Y(H)-1}/d_Y(H)$  // Abhyankar's trick
4.  $\overline{H} \leftarrow \lceil H(X, Y - B) \rceil^N$
5. else  $B \leftarrow 0$ ;  $\overline{H} \leftarrow \lceil H \rceil^N$
6.  $\mathcal{R} \leftarrow \{\}$
7. For  $\Delta$  in  $\mathcal{N}^*(\overline{H})$  do
8.   Compute  $q, m, l$ , and  $\phi_\Delta$
9.    $(a, b) \leftarrow \text{Bézout}(q, m)$
10. For  $(\phi, M)$  in **Factor**( $L_t, \phi$ ) do
11.   Let  $\xi$  be any root of  $\phi$
12.    $\widetilde{N} \leftarrow N/[L_t(\xi) : L_t]$  // Update truncation order

13.  $\overline{H}_{\Delta,\xi} \leftarrow \lceil \overline{H}(\xi^b X^q, X^m(\xi^a + Y))/X^l \rceil^{[\tilde{N}]}$
14.  $\tilde{H}_{\Delta,\xi} \leftarrow \text{WPT}(\overline{H}_{\Delta,\xi}, [\tilde{N}])$
15. For each  $(P, Q)$  in  $\text{ARNP}(L_t(\xi), \tilde{H}_{\Delta,\xi}, \tilde{N})$  do
16.  $C \leftarrow -[B(\xi^b P^q)]^{m d_X(P)+r}$  //  $Q = Q_0 + X^r Y$
17.  $\mathcal{R} \leftarrow \mathcal{R} \cup \{(\xi^b P^q, C + P^m(\xi^a + Q))\}$
18. Return  $\mathcal{R}$ .

**Remark 3.** At line 11, if  $\xi$  has multiplicity one in  $\phi$ , there is no need to execute lines 13 and 14 because the expected output for the recursive call is just  $[X, Y]$ . Similarly, if  $[\tilde{N}] = 0$ , we must have  $\mathcal{I}(\tilde{H}_{\Delta,\xi}) = 1$  and  $\xi$  must have multiplicity one. For the sake of clarity, we have not included this optimization in the description of  $\text{ARNP}$ , but we will take it into account in our complexity analysis because it will simplify intermediate results. For instance, if  $H(0, Y)$  is squarefree, there is no cost for lines  $\geq 11$ .

**Proposition 6.**  $\text{ARNP}(L, F, \mathcal{V}_F)$  returns  $\mathcal{R}_t(F)$ .

*Proof.* (sketch) Truncation orders of line 4, 5 and 13 preserve singular parts because they are the same as in [40]. We just need to check that the two modifications introduced in  $\text{RNPUiseux}$  do not alter the output.

- Consider first Abhyankar's trick. If  $S$  is a Puiseux series for  $H$  and  $(P', Q' = Q'_0 + cX^{r'}Y)$  is the corresponding output of  $\text{ARNP}$ , then  $Q'_0(X) = \lceil S(P') \rceil^{r'}$ . Obviously  $S + B$  is a Puiseux series for  $H(X, Y - B)$ , with regularity index  $r'$  in  $H(X, Y - B)$  and singular part  $\lceil S + B \rceil^{r'}$ . Moreover  $H(X, Y - B)$  and  $\overline{H}$  have the same singular parts. Hence the expected RPE for  $\overline{H}$  is  $(P', \lceil B(P') \rceil^{r'} + Q')$ . But the pair  $(P, Q)$  in line 15 is an RPE for  $\tilde{H}_{\Delta,\xi}$  and  $(P', \lceil B(P') \rceil^{r'} + Q') = (\xi^b P^q, P^m(\xi^a + Q))$  is an RPE for  $\overline{H}$  with regularity index  $r' = m d_X(P) + r$ . Therefore, line 16 correctly compensates line 4.

- For line 14,  $\overline{H}_{\Delta,\xi}$  and its distinguished polynomial  $\widehat{H}_{\Delta,\xi} \in L_t(\xi)[[X]][Y]$  have the same RPEs centered at  $(0, 0)$ . At recursive calls, we are only concerned with RPEs centered at  $(0, 0)$  and we are allowed to discard the other factor. It is shown in [40] that truncation at order  $[\tilde{N}]$  preserves singular parts centered at  $(0, 0)$ . We may thus continue the computation with  $\tilde{H}_{\Delta,\xi}$  instead of  $\overline{H}_{\Delta,\xi}$ .  $\square$

**Example 2.** If  $F \in \mathbb{F}_{29}[X, Y]$  is defined by  $F = \prod_{i=1}^3 (Y - S_i(X)) + X^{19}Y$  with  $S_i = X + X^2 + X^3 + 17X^4 + X^5 + X^6 + X^7 + (-1)^i X^{15/2}$ ,  $1 \leq i \leq 2$  and  $S_3 = X + X^2 + X^3 + X^4$ . We have  $\mathcal{V}_F = 94$  and  $\text{ARNP}$  runs as follow:

- $\mathcal{N}^*(F)$  has a single edge with a unique root. Abhyankar's trick is applied with  $-B = X + X^2 + X^3 + 2X^4 + 20X^5 + 20X^6 + 20X^7$ .
- $\mathcal{N}^*(\overline{H})$  has a single edge  $\Delta 4i + j = 12$  with  $\phi_\Delta = (T - 28)(T - 15)^2$ . We obtain two factors  $H_1 = \tilde{H}_{\Delta,28}$  and  $H_2 = \tilde{H}_{\Delta,15}$ , with respective  $Y$ -degree 1 and 2.
- The recursive call for  $H_1$  returns  $[X, Y]$ .
- Since  $\mathcal{N}^*(H_2)$  has once again a single edge with a unique root, the recursive call for  $H_2$  applies Abhyankar's trick again with  $-B = 10X + 10X^2 + 10X^3 + \dots$ .
- $\mathcal{N}^*(\overline{H}_2)$  has a single edge  $7i + 2j = 14$  and  $\phi_\Delta = T - 1$ . Since  $\mathcal{I}(\overline{H}_2) = 1$ , execution stops at the next recursive call.

Let us now illustrate the reconstruction of RPEs for  $S_1$  and  $S_2$  (lines 16 and 17):

- The terminal call returns  $(P, Q) = [X, Y]$ . Since  $m = 7$ ,  $d_X(P) = 1$ ,  $r = 0$  and  $\xi^b = 1$ , this gives  $C = 10X^2 + 10X^4 + 10X^6$  and a RPE  $[X^2, 10X^2 + 10X^4 + 10X^6 + (Y + 1)X^7]$
- Coming back to the initial call, we have  $r = 7$ ,  $m = 4$ ,  $d_X(P) = 2$ , and  $\xi^b = 1$ . This time we have  $C = X^2 + X^4 + X^6 + 2X^8 + 20X^{10} + 20X^{12} + 20X^{14}$ , which provides the RPE  $[X^2, S_2(X^2) + X^{15}Y]$ . As for  $S_3$ , we have  $r = 0$  and  $d_X(P) = 1$ , which leads to  $C = X + X^2 + X^3 + 2X^4$ , and to the RPE  $[X, S_3(X) + X^4Y]$  ( $\xi = 28$  here).

## 5 Complexity

In this section, our goal is to prove Theorem 1 and 2. We recall that  $F$  is assumed  $Y$ -monic. The following relations are useful and easy to prove:



**Lemma 1.** Consider a function call  $\text{ARNP}(L, F, \mathcal{V}_F)$ . For any input  $(L_t, H, N)$  in the computation tree, we have:

1.  $d_Y(H) = \sum_{\Delta, \xi} d_Y(\tilde{H}_{\Delta, \xi}) q_{\Delta} [L_t(\xi) : L_t]$ .
2.  $Nt = \mathcal{V}_F$ .
3.  $d_Y(H)t \leq d_Y = d_Y(F)$ .

We recall that  $\rho$  denotes the number of  $L$ -RPEs above 0.

**Proposition 7.** The expected number of operations in  $L$  required to factor all characteristic polynomials during the execution of  $\text{ARNP}(L, F, \mathcal{V}_F)$  is in  $\mathcal{O}(\rho d_Y^2 + d_Y^2 \log \mathfrak{q})$ .

*Proof.* We first estimate the cost of a single function call with input  $(L_t, H, N)$ , forgetting for a moment recursive calls. Since  $\text{Factor}(L_t, \phi_{\Delta})$  requires an expected number of  $\mathcal{O}(\deg(\phi_{\Delta})^2 + \deg(\phi_{\Delta}) \log \mathfrak{q}^t)$  operations in  $L_t$ , summing over  $\Delta$ , we get  $\mathcal{O}(d_Y(H)^2 + d_Y(H) \log \mathfrak{q}^t)$  operations in  $L_t$ , hence  $\mathcal{O}(d_Y(H)^2 t + d_Y(H) t^2 \log \mathfrak{q})$  operations in  $L$ . By Lemma 1, this is in  $\mathcal{O}(d_Y^2 + t d_Y \log \mathfrak{q})$ . In order to conclude, we must estimate the sum of these quantities over the computation tree  $\mathcal{T}$ . Let  $R_i$  denote the RPE corresponding to a branch  $\mathcal{B}_i$  of  $\mathcal{T}$ . There are three types of function calls, corresponding to three types of vertices of  $\mathcal{T}$ :

- Type (I):  $\mathcal{N}^*(\overline{H})$  has a single edge with slope  $m/q$  and  $\phi_{\Delta} = \phi^M$ , with  $\phi$  irreducible in  $L_t[T]$ . Two sub-cases may occur:

- Type (I.a):  $d_T(\phi) = 1$ . In this case, thanks to Abhyankar’s trick, we must have  $q > 1$ . Since the product of all integers  $q$  along  $\mathcal{B}_i$  is  $e_i$ , this situation happens at most  $\log_2 e_i$  times along  $\mathcal{B}_i$ .

- Type (I.b):  $d_T(\phi) > 1$ . The product of the degrees of all polynomials  $\phi$  along  $\mathcal{B}_i$  is  $f_i$ , hence this case may occur at most  $\log_2 f_i$  times along  $\mathcal{B}_i$ .

From Proposition 2, we deduce that type (I) calls may occur at most  $\log_2 e_i f_i \leq \log_2(d_Y)$  times along  $\mathcal{B}_i$ . Along  $\mathcal{B}_i$ , all integers  $t$  that occur satisfy  $t \leq f_i$ . Summing costs of type (I) along  $\mathcal{B}_i$ , we get  $\log_2(d_Y) \times \mathcal{O}(d_Y^2 + f_i d_Y \log \mathfrak{q}) = \mathcal{O}(d_Y^2 + f_i d_Y \log \mathfrak{q})$ . Summing over  $i$ , we obtain  $\mathcal{O}(\rho d_Y^2 + d_Y^2 \log \mathfrak{q})$  using Proposition 2.

- Type (II):  $\mathcal{N}^*(\overline{H})$  has several edges, or the characteristic polynomial of the unique edge has several irreducible factors in  $L_t[T]$ . Since algorithm ARNP then separates two groups of RPEs, this can happen at most  $(\rho - 1)$  times. Since these nodes have at least two subtrees, there exists an injective map  $j$  from these nodes to leaves of  $\mathcal{T}$  such that node  $c$  is mapped to leaf  $R_{j(c)}$  of a subtree rooted at  $c$ . With this construction, integer  $t$  associated with  $c$  is at most  $f_{j(c)}$ . Summing costs over all such nodes and using Proposition 2 yields again the expected result.

- Type (III):  $d_Y(H) = 1$ . Those are the leaves of  $\mathcal{T}$  and induce no operations in  $L$ . □

For Theorem 1 to hold, arithmetic operations in a subfield  $L_t$  of a residue field must be performed in  $\mathcal{O}(t)$  operations in  $L$ . Unfortunately, ARNP builds residue fields by adding step by step roots  $\xi_j$  of characteristic polynomials and no  $\mathcal{O}(t)$  algorithm is known if  $L_t$  is represented as a tower of extensions over  $L$  [35, 42]. Following [40], we propose to compute a primitive element and to change the coefficient field representation whenever a new root of a characteristic polynomial is required. To simplify the exposition, transformations related to coefficients fields are not explicitly described in algorithm ARNP, but their complexity must be taken into account. The analysis of [40, Section 5.1] applies to ARNP:

**Proposition 8.** The number of operations in  $L$  required by changes of representation to execute  $\text{ARNP}(L, F, \mathcal{V}_F)$  is in  $\mathcal{O}(\mathcal{V}_F d_Y^2)$ .

**Proposition 9.** Not taking into account univariate factorizations and changes of representation,  $\text{ARNP}(L, F, \mathcal{V}_F)$  requires at most  $\mathcal{O}(\rho d_Y (\mathcal{V}_F + 1))$  operations in  $L$ .

*Proof.* Consider first the execution of one function call, ignoring for now recursive calls.

By Proposition 5, line 2 requires  $\mathcal{O}(d_Y(H))$  operations in  $L_t$ , hence  $\mathcal{O}(t d_Y(H)) \subset \mathcal{O}(d_Y)$  operations in  $L$ .

Shift of line 4 may be performed with  $\mathcal{O}((N+1)d_Y(H))$  operations in  $L_t$ , hence  $\mathcal{O}(t(N+1)d_Y(H)) \subset \mathcal{O}(d_Y(\mathcal{V}_F + 1))$  operations in  $L$ ; see Lemma 1.

Define  $d_{t\xi} = [L_t(\xi) : L_t]$  and consider one execution of line 13. If  $\lfloor \tilde{N} \rfloor > 0$ , then [40, Lemma 2] indicates that it requires  $\mathcal{O}(\tilde{N} d_Y(H))$  operations in  $L_t(\xi)$ , hence  $\mathcal{O}(\tilde{N} d_Y(H) t d_{t\xi})$  operations in  $L$ . Lemma 1 and line 12 gives  $\mathcal{O}(N d_Y) \subset \mathcal{O}(d_Y \mathcal{V}_F)$ . If  $\lfloor \tilde{N} \rfloor = 0$ , Remark 3 indicates that there is no cost at all. If  $s$  denotes the number of pairs  $(\Delta, \xi)$ , total cost for line 13 is thus in  $\mathcal{O}(s d_Y \mathcal{V}_F)$ .

For line 14, the operation count is the same as for line 13.

As for line 16, we denote by  $(P = \lambda X^e, Q)$  an  $L_t(\xi)$ -RPE of  $\tilde{H}_{\Delta, \xi}$  computed recursively and  $r$  (resp.  $f$ ) its regularity index (resp. its residual degree over  $L$ ). Setting  $r' = m e + r$ , the computation of  $\xi^b \lambda^q$  requires less than  $O(f \log d_Y)$  operations in  $L$  and the cost of computing  $C$  is in  $O((r' + 1)f)$  operations in  $L$ . Since  $f$  is the residual degree over  $L$  of an RPE of  $H$ ,  $r' f \leq \mathcal{V}_F$ ; see [40, Proposition 5]. Moreover,  $\sum_{(P, Q)} f \leq t d_Y(H) \leq d_Y$  by Proposition 2. Total cost is thus in  $O(\mathcal{V}_F + d_Y)$ .

Line 17 needs no arithmetic in  $L$  if the output is returned without expanding expressions, except for the computation of  $\xi^a$  and  $\xi^b$ , which can be done in  $O(f \log d_Y(H))$  operations in  $L$ . Summing over  $(P, Q)$  as above, we obtain  $O(d_Y)$ . If an output in expanded form is expected, computations may also be done with  $O((\mathcal{V}_F + 1)d_Y)$  operations in  $L$ .

Altogether, we have shown that a single call to ARNP performs  $O(s d_Y (\mathcal{V}_F + 1))$  operations in  $L$ . We now sum this cost over all nodes of the computation tree  $\mathcal{T}$  following the proof of Proposition 7.

For type (I) function calls,  $s = 1$ . There are at most  $\rho \log_2 d_Y$  of those and the total cost is in  $O(\rho d_Y (\mathcal{V}_F + 1))$ .

For type (II) calls,  $s > 1$  and such a call separate RPEs into  $s$  groups. Consider the tree  $\mathcal{T}'$  where nodes of type (I) are ignored. We set  $s = 0$  for leaves and show by induction on the depth  $\mathcal{D}$  of  $\mathcal{T}'$  that  $\sum_{s \in \mathcal{T}'} s \leq 2\rho - 2$  (with equality when  $\mathcal{T}'$  is a binary tree). For  $\mathcal{D} = 0$ ,  $\mathcal{T}'$  is just a leaf and the formula is correct because  $\rho = 1$ . Assume  $\mathcal{D} > 0$  and let  $s_0$  be the value associated with the root of  $\mathcal{T}'$  (initial call). Removing the root gives  $s_0 \geq 2$  subtrees of lower depth, having respectively  $\rho_1, \dots, \rho_{s_0}$  leaves. The induction hypothesis yields:

$$\sum_{s \in \mathcal{T}'} s = s_0 + \sum_{s \in \mathcal{T}' \setminus \{s_0\}} s \leq s_0 + \sum_{i=1}^{s_0} (2\rho_i - 2) = 2\rho - s_0 \leq 2\rho - 2,$$

and the proposition is proved.  $\square$

**Proof of Theorem 1.** The bound  $\mathcal{V}_F$  can be computed with  $O(d_X d_Y^2)$  operations in  $L$ . Since  $\rho \leq d_Y$  and  $\mathcal{V}_F \leq d_X(2d_Y - 1)$ , the monic case is a direct consequence of Proposition 7, 8 and 9. For non monic  $F$  we follow [40]: algorithm ARNP returns the expected output provided that the truncation bound  $\mathcal{V}_F$  is replaced by  $\mathcal{V}_F + v_X(A_{d_Y})$ , where  $A_{d_Y}(X)$  is the leading coefficient of  $F$ . The complexity analysis must be slightly adapted, but yields the same result.

**Proof of Theorem 2.** The polynomial  $F$  is irreducible in  $\overline{L}[[X]][Y]$  if and only if  $\rho = 1$  and the corresponding RPE has coefficients in  $L$ . This condition is equivalent to the following one: each Newton polygon encountered by ARNP has a unique edge  $\Delta$  and  $\phi_\Delta(T) = (T - \xi)^{d_T(\phi_\Delta)}$ . The latter condition may be tested with the `Oneroot` function at a cost of  $O(d_T(\phi_\Delta)) \subset O(d_Y)$  operations in  $L$ ; see Proposition 5. Hence, it is easy to modify ARNP to abort and return `False` whenever any of these two conditions is not satisfied. The `Oneroot` test will be repeated at most  $\log_2 d_Y$  times, thanks to Abhyankar's trick. There will be no factorization cost, nor change of representation cost. By Proposition 9, execution of the modified algorithm requires  $O(d_Y(\mathcal{V}_F + 1))$  operations in  $L$  because  $\rho = 1$ ; thus Theorem 2 holds.

## 6 Further factorization

In this section, we present a technique that may reduce running times in some cases, but does not improve the worst case complexity bound of Theorem 1. Due to space constraints, all proofs are omitted. The method is based on the following well-known result, that can easily be justified:

**Proposition 10.** Consider  $H \in L_t[X, Y]$ , a  $Y$ -monic polynomial with  $d_Y(H) > 0$  and  $H(X, 0) \neq 0$ . Denote  $\{\Delta_i\}_{1 \leq i \leq u}$  the edges of  $\mathcal{N}(H)$ ,  $-m_i/q_i$  the slope of  $\Delta_i$  (with  $\gcd(m_i, q_i) = 1$ ) and  $\phi_{\Delta_i} = \prod_{j=1}^{c_i} \phi_{ij}^{M_{ij}}$  the factorization  $\phi_{\Delta_i}$  into irreducible elements of  $L_t[T]$ . Then there exists a unique set of  $Y$ -monic  $G_{ij} \in L_t[[X]][Y]$  such that:

1.  $H = \prod_{i=1}^u \prod_{j=1}^{c_i} G_{ij}$ .
2.  $d_Y(G_{ij}) = q_i \deg(\phi_{ij}) M_{ij}$
3.  $\gcd(G_{ij}, G_{i'j'}) = 1$  if  $(i, j) \neq (i', j')$
4.  $\mathcal{N}(G_{ij})$  has a unique edge with slope  $-m_i/q_i$ ,
5. The characteristic polynomial of  $\mathcal{N}(G_{ij})$  is  $\phi_{ij}^{M_{ij}}$ .

Applying line 13 of ARNP to each factor  $[G_{ij}]^N$  for a well-chosen  $N$  instead of  $\overline{H}$  may save useless computations since  $d_Y(G_{ij}) < d_Y(H)$ , provided that an approximate factorization of  $\overline{H}$  that preserves singular parts can be computed at a sufficiently low cost. Since  $\overline{H}(0, Y) = Y^{\deg(\overline{H})}$ , there is no initial factorization that allows to directly construct approximations of the  $G_{ij}$  via Hensel lifting. Algorithm `Split` below explains how to alleviate the problem. But to stay within the complexity bound of Theorem 1, we must first reduce the complexity of Hensel lifting for polynomials with support in a lattice.

**Structured Hensel Lifting.** Let  $(q, m) \in \mathbb{Z}^2$  with  $q > 0$  and  $\gcd(m, q) = 1$  and denote  $\Gamma_{q,m}$  the lattice of  $\mathbb{Z}^2$  generated by  $(0, q)$  and  $(1, m)$ . We also introduce  $L_t[X, Y]_{\Gamma_{q,m}}$ , the ring of polynomials with support in  $\Gamma_{q,m}$ .

**Lemma 2.**  $L_t[X, Y]_{\Gamma_{q,m}} \cap L_t[Y] = L_t[Y^q]$ .

Complexity results in Section 6 are subject to the following hypothesis:

**Hypothesis 1.** *It is possible to multiply two polynomials in  $L_t[X, Y]_{\Gamma_{q,m}}$  with degrees less than  $n_X$  and  $n_Y \geq q$  using at most  $\mathcal{O}(n_X n_Y / q)$  operations in  $L_t$ .*

By notation  $\mathcal{O}(n_X n_Y / q)$ , we mean that, for all  $q \geq 1$  and all  $n_Y \geq q$ , with  $n_Y$  and  $n_X$  sufficiently large, the function is bounded by  $n_X n_Y / q$  times logarithmic factors of  $n_X$  and  $n_Y$ . If  $L_t$  contains sufficiently many roots of unity, the existence of such a multiplication algorithm might be deduced from [34, Section 4.3]. Assuming Hypothesis 1, we get:

**Proposition 11.** *Let  $A, B \in L_t[X, Y]_{\Gamma_{q,m}}$  with  $d_Y(B) \leq d_Y(A) = n_Y$ ,  $q \leq n_Y$ ,  $B$  is  $Y$ -monic and let  $N$  be in  $\mathbb{N}^*$ . There exists an algorithm to compute  $Q, R \in L[X, Y]$  such that  $A = QB + R \pmod{X^N}$  with  $d_Y(R) < d_Y(B)$  requiring no more than  $\mathcal{O}(n_Y N / q)$  operations in  $L$ . Moreover,  $Q$  and  $R$  are in  $L_t[X, Y]_{\Gamma_{q,m}}$ .*

**Proposition 12.** *Let  $H \in L_t[X, Y]_{\Gamma_{q,m}}$  be  $Y$ -monic, assume  $n_Y = d_Y(H) \geq q$  and let  $N$  be in  $\mathbb{N}^*$ . Suppose that  $H(0, Y) = \prod_i h_i(Y)$  with  $h_i \in L_t[Y^q]$  and  $\gcd(h_i, h_j) = 1$  for  $i \neq j$ . Then there exist unique  $H_i \in L_t[X, Y]_{\Gamma_{q,m}}$  such that  $H = \prod_i H_i \pmod{X^N}$  and  $H_i(0, X) = h_i(Y)$ . Moreover, there exists an algorithm `SHensel` such that the function call `SHensel(H, {h_i}_i, q, N)` computes the (ordered) set  $\{H_i\}_i$  with no more than  $\mathcal{O}(n_Y N / q)$  operations in  $L_t$ .*

**Factorization of  $\overline{H}$ .** We can now describe algorithm `Split` to compute an approximate factorization of  $H$  in  $L_t[[X]][Y]$  corresponding to Proposition 10. The following points are essential:

- To get sufficient approximation for the factors, we must start from the edge with greatest slope, i.e., the rightmost edge. Therefore, the classical dichotomic approach used in multi-factor Hensel lifting cannot be applied to reduce complexity further. During a function call, other edges are grouped together and treated recursively; see lines 5 and 9.
- If  $-m/q$  is the slope of the rightmost edge, we use at line 6 a transformation similar to (4) to obtain a polynomial in  $L_t[X, Y]_{\Gamma_{q,m}}$  that allows to use structured Hensel lifting. All factors corresponding to the rightmost edge are computed, together with a factor  $H_0$  corresponding to other edges.
- To get an order  $N$  approximation, we must lift factors up to order  $qN$ ; see [40, Figure 2]. The key point is that the extra factor  $q$  is compensated by the gain given by Proposition 12. Otherwise, this factorization step would worsen our complexity bound because  $q$  may be as large as  $d_Y$ .

`Split`( $L_t, H, N$ )

**Input:**  $L_t$ , a field,  $H \in L_t[X, Y]$ ,  $Y$ -monic, with  $d_Y(H) > 0$  and  $H(X, 0) \neq 0$ ,  $N$  an integer with  $N > v_X(H(X, 0))$ .  
**Output:** A set  $\{(m_i, q_i, H_{ij}, \phi_{ij}, M_{ij})\}_{i,j}$  with  $1 \leq i \leq u$ ,  $1 \leq j \leq c_i$  such that  $H_{ij} = [G_{ij}]^N$  where  $m_i, q_i, G_{ij}, \phi_{ij}$  and  $M_{ij}$  are defined in Proposition 10.

1. Compute the quantities  $m, q, l, \phi_\Delta$  associated with the rightmost edge  $\Delta$  of  $\mathcal{N}(H)$ .
2.  $\{(\phi_j, M_j)\}_{j=1}^c \leftarrow \text{Factor}(L_t, \phi_\Delta)$
3. Let  $(i_0, j_0)$  be the leftmost point of  $\Delta$ .
4. If  $i_0 = 0$  and  $c = 1$  then Return  $\{(m, q, H, \phi_1, M_1)\}$ .
5. If  $i_0 > 0$  then

Write  $i_0 = (a - 1)q + b$  with  $0 < b \leq q$  and set  $r = q - b$ .  
 $(\phi_0, M_0) \leftarrow (T^a, 1)$

*// There is more than one edge.*

6.  $\widehat{H}(X, Y) \leftarrow Y^r H(X^q, X^m Y) / X^l \in L[X, Y]$ .
7.  $\{\widehat{H}_j\}_j \leftarrow \text{SHense1}(\widehat{H}, \{\phi_j(Y^q)^{M_j}\}_j, q, qN + 1)$
8. For  $j$  from 1 to  $c$  do //  $\widehat{H}_j$  corresponds to  $\phi_j$ .  
 $H_j(X, Y) \leftarrow \widehat{H}_j(X^{1/q}, X^{-m/q} Y) X^{m \deg(\phi_j) M_j}$
9. If  $i_0 > 0$  then // Treat remaining edges.  
 $H_0(X, Y) \leftarrow \widehat{H}_0(X, Y) X^{m a} Y^{-r}$   
 $S \leftarrow \text{Split}(L_t, H_0, N)$
10. Return  $S \cup \{(m, q, H_j, \phi_j, M_j)\}_{1 \leq j \leq c}$

Note that at line 7, index  $j$  ranges from 0 to  $c$  if  $i_0 > 0$  and from 1 to  $c$  if  $i_0 = 0$ .

**Proposition 13.** *Algorithm Split returns the expected output. Not taking into account operations induced by sub-algorithm Factor, it requires at most  $\mathcal{O}(uNd_Y(H))$  operations in  $L_t$ , where  $u$  is the number of edges of  $\mathcal{N}(H)$ .*

Function ARNP may be easily modified to include this factorization step: After line 5, include a line

5b.  $\{(m_k, q_k, H_k, \phi_k, M_k)\}_{1 \leq k \leq s} \leftarrow \text{Split}(L, \overline{H}, \lfloor N \rfloor)$ ,

then replace the nested ‘‘For’’ loops over  $\Delta$  and  $\xi$  by a loop over the  $(m_k, q_k, H_k, \phi_k, M_k)$  for  $1 \leq k \leq s$ , and continue the processing as before. The modified algorithm must also take a special care of the first edge of  $\overline{H}$ , if the corresponding  $H_k$  has degree 1, otherwise it may return expansions with superfluous terms. This causes no significant problem and has no impact on the complexity, hence we omit these technical details.

From the proof of Proposition 9, cost for lines 13 and 14 of one function call is  $\mathcal{O}(sNd_Y(H))$  operations in  $L_t$ , where  $s$  is the number of pairs  $(\Delta, \xi)$ . With the above modifications, this becomes  $\mathcal{O}(uNd_Y(H)) + \sum_{k=1}^s \mathcal{O}(Nd_Y(H_k)) \subset \mathcal{O}(uNd_Y(H))$  because  $\sum_k d_Y(H_k) = d_Y(H)$ . This factorization step is thus worthwhile if  $u/s$  is sufficiently small to compensate larger factors hidden by the notation  $\mathcal{O}$ .

## 7 Conclusion

Theorem 1 reduces by one order of magnitude the bound of [40] for the computation of RPEs of  $F$  above 0. Example 3 below shows that our operation count is sharp because Algorithm ARNP requires  $\Theta(d^4)$  operations in  $L$  for this case.

**Example 3.** *Define  $S_k = 2X^k + \sum_{l=1}^{k-1} X^l$  and  $F(X, Y) = \prod_{k=1}^N (Y - S_k)$ . At each call of ARNP,  $H$  has single edge  $i + j = d_Y(H)$ , with  $\phi_\Delta = (T - 2)(T - 1)^{d_Y(H)-1}$  and Abhyankar’s trick does not save any function call. We have  $d_Y = N$ ,  $d_X = \frac{N(N+1)}{2}$  and  $\mathcal{V}_F = \frac{(N-1)N(N+1)}{3} \in \Theta(d_X d_Y)$ . Moreover, ARNP will execute  $\rho - 2 = d_Y - 2$  recursive calls; this leads to a complexity in  $\Theta(d_X d_Y^3)$ .*

It turns out that the  $\mathcal{O}(d^5 \log q)$  complexity bound derived in [40] using the Riemann-Hurwitz formula for the computation of the genus of the curve  $F(X, Y) = 0$  cannot be decreased by a mere application of Theorem 1. Indeed, suppose that  $\Delta_F$  has a large irreducible factor  $D$  in  $L[X]$  of degree  $t_0$  close to  $d_X d_Y$ . In order to apply the Riemann-Hurwitz formula, we need to compute RPEs above 0 of  $F_c(X, Y) = F(X + c, Y) \in L_{t_0}[X, Y]$  where  $c$  is a root of  $D$ . When applying ARNP to  $F_c$ , if a characteristic polynomial  $\phi$  of degree close to  $d_Y$  is encountered, the factorization of  $\phi$  in  $L_{t_0}[T]$  alone will require  $\mathcal{O}(d_Y^2 + d_Y \log q^{t_0})$  operations in  $L_{t_0}$  with standard factorization algorithms, thus  $\mathcal{O}(d^5 \log q)$  operations in  $L$ . Unless a univariate factorization algorithm with a drastically reduced running time is discovered, there is no hope to get an  $\mathcal{O}(d^4 \log q)$  bound with this method (the recent algorithm of [30] is not even sufficient). However, following [40], we obtain:

**Proposition 14.** *Not taking into account univariate factorizations, there exists an algorithm to compute the genus of the curve  $F(X, Y) = 0$  with  $\mathcal{O}(d_X d_Y^2 (d_X + d_Y)) \subset \mathcal{O}(d^4)$  operations in  $L$ .*

This result suggests to use the D5 technique [20, 18] to avoid the univariate factorization bottleneck; this will be the topic of forthcoming investigations.

A prototype for ARNP has been implemented in Maple to validate the algorithm, but a significant amount of work is still necessary to develop efficient code. In fact, the technique introduced in this paper give better asymptotic

operation counts and better upper complexity bounds, but it is not even clear that ARNP can be made to run faster than other implementations for reasonable input size. In particular, the truncation bound  $\mathcal{V}_F$  is usually not sharp (as demonstrated by Example 2 where the value  $\mathcal{V}_F = 94$  is far from optimal) and calls to WPT artificially increase  $X$ -degree. Finer bounds, or/and a “relaxed” approach [48] could prove useful.

As for Section 6, we consider it for now as a motivation for studying further structured multiplication algorithms [34]. No implementation of those is known to the authors and this would be a significant contribution in itself.

**Acknowledgments.** We are grateful to Eric Schost for useful input about [34]. We thank anonymous referees for their contribution to the clarity of the paper.

## References

- [1] S. Abhyankar. *Algebraic Geometry for Scientists and Engineers*, volume 35 of *Mathematical surveys and monographs*. Amer. Math. Soc., 1990.
- [2] J.-D. Bauch. Genus computation of global function fields. *Journal of Symbolic Computation*, 66:8–20, 2015.
- [3] J.-D. Bauch, E. Nart, and H. Stainsby. Complexity of the OM factorizations of polynomials over local fields. *LMS Journal of Computation and Mathematics*, 16:139–171, 2013.
- [4] T. Beck. Formal desingularization of surfaces: The Jung method revisited. *J. Symb. Comp.*, 44(2):131–160, 2009.
- [5] J. Berthomieu, G. Lecerf, and G. Quintin. Polynomial root finding over local rings and application to error correcting codes. *Applicable Algebra in Engineering, Communication and Computing*, 24(6):413–443, 2013.
- [6] D. Bini and V. Y. Pan. *Polynomial and Matrix Computations*, volume 1 of *Progress in Theoretical Computer Science*. Birkhäuser, Saarbrücken, 1994.
- [7] G. A. Bliss. *Algebraic functions*. AMS, 1933.
- [8] W. Bosma, J. Cannon, and C. Playoust. The Magma Algebra System I : The user language. *J. Symb. Comp.*, 24(3-4):235–265, 1997.
- [9] E. Brieskorn and H. Knörrer. *Plane Algebraic Curves*. Birkhäuser, 1986.
- [10] A. Campillo. *Algebroid Curves in Positive Characteristic*, volume 378 of *LNCS*. Springer-Verlag, 1980.
- [11] C. Chevalley. *Introduction to the Theory of Algebraic Functions of One Variable*, volume 6 of *Mathematical Surveys*. AMS, 1951.
- [12] A. L. Chistov. Polynomial complexity of the Newton-Puiseux algorithm. In *Mathematical Foundations of Computer Science 1986*, pages 247–255, London, UK, 1986. Springer-Verlag.
- [13] D. V. Chudnovsky and G. V. Chudnovsky. On expansion of algebraic functions in power and puiseux series. I. *Journal of Complexity*, 2(4):271–294, 1986.
- [14] D. V. Chudnovsky and G. V. Chudnovsky. On expansion of algebraic functions in power and puiseux series. II. *Journal of Complexity*, 3(1):1–25, 1987.
- [15] P. M. Cohn. Puiseux’s Theorem revisited. *Journal of Pure and Applied Algebra*, 24:1–4, 1984.
- [16] L. Comtet. Calcul pratique des coefficients de Taylor d’une fonction algébrique. *L’Enseignement Mathématique*, 2(10):267–270, 1964.
- [17] O. Cormier, M. F. Singer, B. M. Trager, and F. Ulmer. Linear differential operators for polynomial equations. *J. Symb. Comp.*, 34(5):355–398, 2002.
- [18] X. Dahan, E. Schost, M. M. Maza, W. Wu, and Y. Xie. On the complexity of the D5 principle. *SIGSAM Bull.*, 39(3):97–98, 2005.
- [19] W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann. SINGULAR 3-1-6 — A computer algebra system for polynomial computations. <http://www.singular.uni-kl.de>, 2012.
- [20] J. Della Dora, C. Dicrescenzo, and D. Duval. About a new method for computing in algebraic number fields. In

- [21] G. Diaz-Toca and L. Gonzalez-Vega. Determining puiseux expansions by hensel's lemma and dynamic evaluation. In V. Ganzha, E. Mayr, and E. Vorozhtsov, editors, *Computer Algebra in Scientific Computing, CASC 2002*. Technische Universität München, Germany, Sept. 2002.
- [22] D. Duval. Diverses questions relatives au calcul formel avec des nombres algébriques. Université de Grenoble, Thèse d'État, 1987.
- [23] D. Duval. Rational Puiseux expansions. *Compositio Math.*, 70(2):119–154, 1989.
- [24] M. Eichler. *Introduction to the Theory of Algebraic Numbers and Functions*. Academic Press, 1966.
- [25] D. Ford and O. Veres. On the complexity of the Montes ideal factorization. In Springer, editor, *ANTS IX*, volume 6197, pages 174–185. Lecture Notes in Computer Science, 2010.
- [26] G.-M. Greuel, C. Lossen, and M. Schulze. Three algorithms in Algebraic Geometry, Coding Theory and Singularity Theory. In Kluwer, editor, *Application of Algebraic Geometry to Coding Theory, Physics and Computation*, pages 161–194, 2001.
- [27] J.-P. Henry and M. Merle. Complexity of computation of embedded resolution of algebraic curves. In *Proceedings Eurocal 87*, number 378 in Lecture Notes in Computer Science, pages 381–390. Springer-Verlag, 1987.
- [28] D. Inaba. Factorization of multivariate polynomials by extended hensel construction. *SIGSAM Bull.*, 39(1):142–154, 2005.
- [29] K. S. Kedlaya. The algebraic closure of the power series fields in positive characteristic. *Proc. Amer. Math. Soc.*, 129:3461–3470, 2001.
- [30] K. S. Kedlaya and C. Umans. Fast polynomial factorization and modular composition. *SIAM J. Computing*, 40(6):1767–1802, 2011.
- [31] H. T. Kung and J. F. Traub. All algebraic functions can be computed fast. *J. ACM*, 25(2):245–260, 1978.
- [32] T.-C. Kuo. Generalized Newton-Puiseux Theory and Hensel's Lemma in  $c[[x, y]]$ . *Can. J. of Math.*, XLI:1101–1116, 1989.
- [33] T.-C. Kuo. A simple algorithm for deciding primes in  $k[[x, y]]$ . *Can. J. of Math.*, 47(4):801–816, 1995.
- [34] J. Lebreton, E. Schost, and J. V. der Hoeven. Structured FFT and TFT: symmetric and lattice polynomials. In ACM, editor, *Proc. ISSAC '13*, pages 355–362, 2013.
- [35] X. Li, M. M. Maza, and E. Schost. Fast arithmetic for triangular sets: from theory to practice. In *Proc. ISSAC '07*, pages 269–276, New York, NY, USA, 2007. ACM.
- [36] M. B. Monagan, K. O. Geddes, K. M. Heal, G. Labahn, S. M. Vorkoetter, J. McCarron, and P. DeMarco. *Maple 10 Programming Guide*. Maplesoft, Waterloo ON, Canada, 2005.
- [37] S. Pauli. Factoring polynomials over local fields. *J. Symb. Comp.*, 32(533-547), 2001.
- [38] S. Pauli. Factoring polynomials over local fields ii. In Springer, editor, *ANTS IX*, volume 6197 of *Lecture Notes in Computer Science*, pages 301–315, 2010.
- [39] A. Poteaux and M. Rybowicz. Good reduction of puiseux series and complexity of the Newton-Puiseux algorithm. In *Proc. ISSAC '08*, pages 239–246, New-York, 2008. ACM.
- [40] A. Poteaux and M. Rybowicz. Complexity Bounds for the Rational Newton-Puiseux Algorithm over Finite Fields. *Applicable Algebra in Engineering, Communication and Computing*, 22(3):187–217, 2011.
- [41] A. Poteaux and M. Rybowicz. Good reduction of Puiseux series and applications. *J. Symb. Comp.*, 47(1):32–63, 2012.
- [42] A. Poteaux and E. Schost. On the complexity of computing with zero-dimensional triangular sets. *J. Symb. Comp.*, 50:110–138, 2013.
- [43] M. Rybowicz. An algorithm for computing an integral basis in an algebraic function field. In *Proc. ISSAC '91*. ACM Press, 1991.

- [44] T. Sasaki and D. Inaba. Hensel construction of  $f(x, u_1, \dots, u_l)$  at a singular point and its application. *SIGSAM Bull.*, 1:9–17, 2000.
- [45] T. Sasaki and F. Kako. Solving multivariate algebraic equations by Hensel construction. *Japan J. of Industrial and Applied Math.*, 16:257–285, 1999.
- [46] J. Teitelbaum. The computational complexity of the resolution of plane curve singularities. *Math. Comp.*, 54(190):797–837, 1990.
- [47] J. van der Hoeven. Fast evaluation of holonomic functions. *Theoretical Computer Science*, 210(1):199–215, 1999.
- [48] J. van der Hoeven. Relax, but don't be too lazy. *J. Symb. Comp.*, 34:479–542, 2002.
- [49] J. van der Hoeven. Effective analytic functions. *J. Symb. Comp.*, 39(3-4):433–449, 2005.
- [50] M. van Hoeij. An algorithm for computing an integral basis in an algebraic function field. *J. Symb. Comp.*, 18:353–363, 1994.
- [51] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, Cambridge, 1999.
- [52] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, second edition, 2003.
- [53] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, third edition, 2013.
- [54] R. J. Walker. *Algebraic Curves*. Springer Verlag, Berlin-New York, 1978.
- [55] P. G. Walsh. On the complexity of rational Puiseux expansions. *Pacific J. of Math.*, 188:369–387, 1999.
- [56] P. G. Walsh. A polynomial-time complexity bound for the computation of the singular part of an algebraic function. *Math. of Comp.*, 69:1167–1182, 2000.

## 8 Appendix

We present proofs omitted in the paper.

*of Proposition 3.* The existence and unicity of  $U \in L_t[[X, Y]]$  and  $\widehat{G}$  is guaranteed by Weierstrass Preparation Theorem. Writing  $U = \sum_i U_i(Y)X^i$ , it is a simple matter to show recursively that  $U_i$  is a polynomial with  $d_Y(U_i) \leq d_Y(G) - \mathcal{I}(G)$ . Finally, it is easily seen that Puiseux series  $S$  of  $\widehat{G}$  and  $G$  that vanish at  $X = 0$  are the same.  $\square$

*of Proposition 4 (sketch).* Define  $u = \mathcal{I}(G)$  and  $h(Y) = Y^u$ . If  $N = 0$ , just return  $h$  and there is no operation in  $L_t$  to execute. If  $N > 0$ , we apply the quadratic Hensel lifting technique described in [51, Chapter 15] to compute  $\widetilde{G}$ . Set  $g(Y) = G(0, Y)/Y^u$ , so that  $G = gh \pmod{X}$ , and compute  $s, t \in L_t[Y]$  satisfy  $sg + th = 1$  using  $\mathcal{O}(d_Y)$  operations in  $L_t$ . Hypotheses of [51, Algorithm 15.10], that performs a Hensel lifting step, are not verified by polynomials  $G, g, h, s$  and  $t$  because the leading coefficient of  $G$  is not invertible in  $L_t[[X]]$  and  $d_Y(g) + d_Y(h)$  may be less than  $d_Y(G)$ . However, an examination of the proof of this algorithm ([51, Theorem 15.11]) shows that it can still be applied. Therefore, [51, Algorithm 15.17] computes  $\widetilde{g}$  and  $\widetilde{h}$  in  $L_t[X, Y]$  such that  $G = \widetilde{g}\widetilde{h} \pmod{X^{N+1}}$ ,  $\widetilde{g} = g \pmod{X}$  and  $\widetilde{h} = h \pmod{X}$ , using  $\mathcal{O}(Nd_Y(G))$  operations in  $L_t$ . It is possible to adapt proof of [51, Theorem 15.14] to show that  $\widetilde{g}$  and  $\widetilde{h}$  are the unique polynomials satisfying these relations. Since  $\widehat{G}(0, Y) = h$  and  $U(0, Y) = g$ , we must have  $\widetilde{h} = \lceil \widehat{G} \rceil^N$ .  $\square$

*of Lemma 1.* Point 1 is a direct consequence of the definition of  $\phi_\Delta$ , noting that the multiplicity  $M_{\Delta, \xi}$  of  $\xi$  in  $\phi_\Delta$  is equal to  $d_Y(\widetilde{H}_{\Delta, \xi})$ . Equality 2 is easily proved by induction, since  $N$  is divided by  $[L_t(\xi) : L_t]$  at each function call, and  $t$  multiplied by the same quantity. Statement 3 can also be proved recursively using 1.  $\square$

*of Proposition 5.* Writing  $\phi(T) = \sum_{i=0}^n a_i T^i$ , it is sufficient to test whether the shifted polynomial  $\phi(T - a_{n-1}/a_n/n)$  is equal to  $a_n T^n$ , at a cost of  $\mathcal{O}(n)$ ; see assumptions in Section 1.  $\square$

of Proposition 8. We refer the reader to [40, Section 5.1] and use the same notations: bounds given for steps (A), (B), (D) are explicitly in  $\mathcal{O}(\delta_F d_Y) \subset \mathcal{O}(\mathcal{V}_F d_Y)$ . As for step (C), the last paragraph of [40, Section 5.1.3] indicates that it can be done with  $\mathcal{O}(\mathcal{V}_F d_Y^2)$  operations in  $L$ .  $\square$

of Proposition 10 (sketch). Let  $\xi_{ij}$  be a root of  $\phi_{ij}$ . Then  $G_{ij}$  is just the product of the minimal polynomials over  $L_t[[X]]$  of Puiseux series of  $H$  above 0 (counted with multiplicities if  $H$  is not squarefree) with first term equal to  $\xi_{ij} X^{m_i/q_i}$ ; see factorization (1) with  $F = H$ . All properties are then easily checked since  $M_{ij}$  is the number of such minimal polynomials (counted with multiplicities) and  $q_i \deg(\phi_{ij})$  their degrees.  $\square$

of Proposition 11. To compute  $Q$  and  $R$ , we follow [52, Algorithm 9.5]. Let  $a = n_Y$  and  $b$  denote respectively degrees in  $Y$  of  $A$  and  $B$ . For any polynomial  $H \in L_t[X, Y]_{\Gamma_{q,m}}$ , we denote  $\tilde{H}(X, Y) = Y^{d_Y(H)} H(X, 1/Y)$  the reciprocal polynomial of  $H$ . We assume  $b > 0$ , otherwise, the result is trivial. Since  $B$  is monic, Lemma 2 gives  $b = qk$  for some  $k \in \mathbb{N}^*$ . Therefore, a monomial of  $\tilde{B}$  has the form  $X^{-m(-i+kq)+q(j+km)} Y^{-i+kq}$  for some  $(i, j) \in \mathbb{Z}^2$ , which shows that  $\tilde{B}$  is in  $L_t[X, Y]_{\Gamma_{q,-m}}$ . The inverse  $\tilde{B}^{-1} \pmod{(Y^{(b-a+1)}, X^N)}$  may then be computed via Newton's iterations using only multiplications and additions in  $L_t[X, Y]_{\Gamma_{q,-m}}$ . If  $a$  is also a multiple of  $q$ , then  $\tilde{A}$  is also in  $L_t[X, Y]_{\Gamma_{q,-m}}$  and the computation of

$$\tilde{Q} = \tilde{A} \tilde{B}^{-1} \pmod{(Y^{(b-a+1)}, X^N)}$$

is essentially a product in  $L_t[X, Y]_{\Gamma_{q,-m}}$ . Hence, it is sufficient to replace in [52, Algorithm 9.5] bivariate multiplications by structured multiplications of Hypothesis 1 to obtain an algorithm that needs at most  $\mathcal{O}(n_X n_Y / q)$  operations in  $L_t$ . Moreover,  $Q$  and  $R$  are in  $L_t[X, Y]_{\Gamma_{q,m}}$  because  $L_t[X, Y]_{\Gamma_{q,m}}$  is a ring. When  $a$  is not a multiple of  $q$ ,  $\tilde{A}$  may not be in  $\Gamma_{q,-m}$ , but the following workaround solves the problem: Let  $a'$  be the smallest multiple of  $q$  greater than  $a$  and set  $A_1 = Y^{a'} + A$ ,  $A_2 = Y^{a'}$ . We may now compute quotients and remainders for the  $A_i$  in  $\mathcal{O}(a' n_X / q)$  operation in  $L$ . Since  $a' < a + q$  and  $a \geq b \geq q$  because  $b$  is a non zero multiple of  $q$ ,  $a' \leq 2a$  and we get the expected bound. Finally, set  $Q = Q_1 - Q_2$  and  $R = R_1 - R_2$ .  $\square$

of Proposition 12 (sketch). Noting that  $H$  is monic, the existence, unicity and computability of the  $H_i$  are guaranteed by the Hensel lifting algorithm [52, Chapter 15]. We are left to prove the complexity estimate and the fact that  $H_i$  is in  $L_t[X, Y]_{\Gamma_{q,m}}$ . When  $q = 1$ , this complexity bound is well known; see [52, Theorem 15.18]. We argue as in the proof of Proposition 11: An examination of Hensel multi-factor lifting algorithm, as presented in [52, Algorithm 15.17], reveals that it boils down to additions, multiplications and Euclidean division modulo powers of  $X$  (where divisors are always monic in  $Y$ ) of polynomials in  $L_t[X, Y]_{\Gamma_{q,m}}$ . The cost of each elementary operation may thus be divided by  $q$  and this gives the expected complexity.  $\square$

of Proposition 13. If  $i_0 = 0$  and  $c = 1$ , **Split** obviously returns the expected result. Next, note that a monomial  $X^j Y^i$  is transformed by line 6 into  $X^{mi+qj-l} Y^{i+r}$ . Let us show that the corresponding affine application defined by  $\Psi(i, j) = (i, mi + qj) + (r, -l)$  is a bijection from  $\mathbb{Z}^2$  to  $\Gamma_{q,m}$ . Since  $(i_0, j_0)$  belongs to  $\Delta$ ,  $l = mi_0 + qj_0$  and  $(r, -l) = (r, -mi_0 - qj_0) = (r, m(r - aq) - qj_0) = r(1, m) - (ma + j_0)(0, q) \in \Gamma_{q,m}$ , which proves that  $\Psi(i, j)$  belongs to  $\Gamma_{q,m}$ . Finally,  $\Psi$  is bijective because its matrix is unimodular. We deduce that  $\hat{H}$  belongs to  $L[X, Y]_{\Gamma_{q,m}}$ . For  $v \in \mathbb{N}$ , we denote by  $\mathcal{L}_v$  the line  $mi + qj = l + v$ . It is easily checked that  $\Psi(\mathcal{L}_v)$  is the horizontal of ordinate  $v$ ; see [40, Figure 2]. For  $v = 0$ , this shows that 1) by Lemma 2,  $\hat{H}(0, Y)$  belongs to  $L[Y^q]$ , 2)  $\hat{H}$  is monic in  $Y$  and 3)  $\mathcal{I}(\hat{H}) = i_0 + r = aq$ . From the definition of  $\phi_\Delta$  and lines 2 and 5, we get:  $\hat{H}(0, Y) = \prod_j \phi_j(Y^q)^{M_j}$ , where the product ranges from 0 to  $c$  if  $i_0 > 0$  and from 1 to  $c$  if  $i_0 = 0$ . Since the  $\phi_j(Y^q)$  are pairwise coprime, algorithm **SHensel** is correctly initialized and returns polynomials satisfying:

$$\hat{H} = \prod_j \hat{H}_j \pmod{X^{qN+1}}. \quad (5)$$

Consider the factorization of  $H$  in Proposition 10 and define  $G_{u0} = \prod_{i=1}^{u-1} \prod_{j=1}^{c_i} G_{ij}$ ,  $\hat{G}_{u0} = Y^r G_0(X^q, X^m Y) / X^{md_Y(G_{u0})}$  and  $\hat{G}_{uj} = G_{uj}(X^q, X^m Y) / X^{md_Y(G_{uj})}$  for  $1 \leq j \leq c_u$ . The  $\hat{G}_{uj}$  satisfy  $\hat{H} = \prod_j \hat{G}_{uj}$  because  $l = md_Y(H)$  and  $d_Y(H) = \sum_j d_Y(G_{uj})$ . Proceeding as above, it can be shown that the  $\hat{G}_{uj}$  are in  $L[[X]][Y]_{\Gamma_{q,m}}$ . Up to a re-indexing, we have  $\hat{G}_{uj}(0, Y) = \phi_j(Y^q)^{M_j} = \hat{H}_j(0, Y)$ . From the unicity of the  $\hat{H}_j$  in (5), we get:

$$\hat{H}_j = \hat{G}_{uj} \pmod{X^{qN+1}}, \quad 0 \leq j \leq c_u.$$



Applying reverse transformations, we get  $Y$ -monic polynomials  $H_j$  that verify:

$$H_j = G_{uj} \pmod{\mathcal{L}_{qN+1}}, \quad 0 \leq j \leq c_u,$$

where, for any  $U, V \in L_t[X, Y]$ ,  $U = V \pmod{\mathcal{L}_v}$  means that  $U - V$  is in the ideal generated by the monomials with support in  $\mathcal{L}_v$ . Consider now the intersection of  $\mathcal{L}_{qN+1}$  with the vertical line of abscissa  $d_Y(H_j)$ . The ordinate  $j_1$  of this point satisfies  $md_Y(H_j) + qj_1 = qN + l + 1$ , hence  $j_1 = N + (l - md_Y(H_j) + 1)/q = N + m(d_Y(H) - d_Y(H_j) + 1)/q \geq N$ . Hence,  $H_j$  contains all monomials of  $G_{uj}$  up to  $X$ -degree  $N$  and we get:

$$H_j = G_{uj} \pmod{X^{N+1}} \text{ and } H = \prod_j H_j \pmod{X^{N+1}}.$$

In particular, if  $i_0 \neq 0$ , we have  $N > v_X(H(X, 0)) \geq v_X(H_0(X, 0))$ ,  $H_0(X, 0) \neq 0$  and  $d_Y(H_0) > 0$ . Algorithm `Split` may thus be applied recursively to  $H_0$ .

Finally, note that algorithm `SHensel` is called at most  $u$  times. Proposition 12 then gives the complexity bound.  $\square$